CLAIMS

What is claimed is:

1 1	1	4 meth	rod co	ompris	ing
-----	---	--------	--------	--------	-----

- 2 running guest software in a processor mode that enables the guest
- 3 software to operate at a privilege level intended by the guest software; and
- 4 responsive to an attempt of the guest software to perform an operation
- 5 restricted by said processor mode, exiting said processor mode to transfer
- 6 control over the operation to the VMM running outside said processor mode.
- 1 2. The method of claim 1 further comprising:
- 2 responding to the operation; and
- 3 transferring control over the operation to the guest software by
- 4 entering said processor mode.
- 1 3. The method of claim 2 wherein entering said processor mode includes
- 2 loading processor state expected by the guest software.
- 1 4. The method of claim 1 wherein exiting said processor mode further
- 2 comprises:
- 3 saving processor state used by the guest software; and
- 4 loading processor state required by the VMM.

- 1 5. The method of claim 1 wherein exiting said processor mode further
- 2 comprises automatically transferring from an address space associated with
- 3 the guest software to an address space associated with the VMM.
- 1 6. The method of claim 1 further comprising maintaining a flag in a
- 2 processor control register to indicate whether the processor is in said
- 3 processor mode.
- 1 7. The method of claim 1 further comprising reporting an ability of a
- 2 processor to support said processor mode using one of a plurality of reserved
- 3 feature bits that are returned in a processor register.
- 1 8. The method of claim 1 wherein exiting said processor mode comprises
- 2 generating one of a plurality of interrupts and exceptions in response to the
- 3 attempt of the guest software to perform the operation restricted by said
- 4 processor mode.
- 1 9. The method of claim 8 wherein generating one of the plurality of
- 2 interrupts and exceptions further includes:
- identifying the attempt of the guest software to perform the operation
- 4 restricted by said processor mode; and
- 5 determining that the attempt of the guest software is potentially
- 6 successful.

- 1 10. The method of claim 8 further comprising:
- 2 maintaining a redirection bitmap for the plurality of the interrupts and
- 3 exception, the redirection bitmap indicating whether each of the plurality of
- 4 the interrupts and exceptions is allowed to be handled by the guest software;
- 5 and
- 6 consulting the redirection bitmap to determine whether to exit said
- 7 processor mode.
- 1 11. The method of claim 8 further comprising:
- 2 identifying an attempt of the guest software to modify an interrupt
- 3 flag; and
- 4 modifying the interrupt flag if the interrupt flag does not control
- 5 masking of interrupts.
- 1 12. The method of claim 8 further comprising:
- identifying an attempt of the guest software to modify an interrupt
- 3 flag; and
- 4 preventing the attempt of the guest software to modify the interrupt
- 5 flag.
- 1 13. The method of claim 12 wherein preventing the attempt of the guest
- 2 software to modify the interrupt flag includes providing a shadow interrupt
- 3 flag for modifications by the guest software.

- 1 14. The method of claim 12 wherein preventing the attempt of the guest
- 2 software to modify the interrupt flag includes generating one of the plurality
- 3 of interrupts and exceptions in response to the attempt of the guest software
- 4 to modify the interrupt flag.
- 1 15. A system comprising:
- 2 a memory; and
- a processor, coupled to the memory, to run guest software in a
- 4 processor mode that enables the guest software to operate at a privilege level
- 5 intended by the guest software, to identify an attempt of the guest software to
- 6 perform an operation restricted by said processor mode, and to exit said
- 7 processor mode, in response to the attempt, to transfer control over the
- 8 operation to a virtual-machine monitor (VMM) running outside said
- 9 processor mode.
- 1 16. The system of claim 15 wherein the processor is to re-enter said
- 2 processor mode after the VMM responds to the operation.
- 1 17. The system of claim 16 wherein the processor is to load processor state
- 2 expected by the guest software when re-entering said processor mode.
- 1 18. The system of claim 15 wherein the processor is to save processor state
- 2 used by the guest software and to load processor state required by the VMM
- 3 when exiting said processor mode.

- 1 19. The system of claim 15 wherein exiting said processor mode further
- 2 comprises automatically transferring from an address space associated with
- 3 the guest software to an address space associated with the VMM.
- 1 20. The system of claim 15 wherein the processor is to maintain a flag in a
- 2 processor control register to indicate whether the processor is in said
- 3 processor mode.
- 1 21. The system of claim 15 wherein the processor is to reporting an ability
- 2 to support said processor mode using one of a plurality of reserved feature
- 3 bits that are returned in a processor register.
- 1 22. The system of claim 15 wherein the processor is to generate one of a
- 2 plurality of interrupts and exceptions in response to the attempt of the guest
- 3 software to perform the operation restricted by said processor mode.
- 1 23. The system of claim 22 wherein the processor is to generate one of the
- 2 plurality of interrupts and exceptions upon determining that the attempt of
- 3 the guest software to perform the operation restricted by said processor mode
- 4 is potentially successful.
- 1 24. The system of claim 22 wherein the processor is to consult a redirection
- 2 bitmap to determine whether to exit said processor mode, the redirection

- 3 bitmap indicating whether each of the plurality of the interrupts and
- 4 exceptions is allowed to be handled by the guest software.
- 1 25. The system of claim 22 wherein the processor is to identify an attempt
- 2 of the guest software to modify an interrupt flag and to modify the interrupt
- 3 flag if the interrupt flag does not control masking of interrupts.
- 1 26. The system of claim 22 wherein the processor is to identify an attempt
- 2 of the guest software to modify an interrupt flag and to prevent the attempt of
- 3 the guest software to modify the interrupt flag.
- 1 27. The system of claim 26 wherein the processor is to prevent the attempt
- 2 of the guest software to modify the interrupt flag by providing a shadow
- 3 interrupt flag for modifications by the guest software.
- 1 28. A computer readable medium that provides instructions, which when
- 2 executed on a processor, cause said processor to perform operations
- 3 comprising:
- 4 running guest software in a processor mode that enables the guest
- 5 software to operate at a privilege level intended by the guest software; and
- 6 responsive to an attempt of the guest software to perform an operation
- 7 restricted by said processor mode, exiting said processor mode to transfer
- 8 control over the operation to the VMM running outside said processor mode.

- 1 29. The computer readable medium of claim 28 providing further
- 2 instructions causing the processor to perform operations comprising:
- 3 responding to the operation; and
- 4 transferring control over the operation to the guest software by
- 5 entering said processor mode.
- 1 30. The computer readable medium of claim 28 comprising further
- 2 instructions causing the processor to perform operations comprising:
- 1 maintaining a redirection bitmap for the plurality of the interrupts and
- 2 exception, the redirection bitmap indicating whether each of the plurality of
- 3 the interrupts and exceptions is allowed to be handled by the guest software;
- 4 and
- 5 consulting the redirection bitmap to determine whether to exit said
- 6 processor mode.